Real Time Tracking of High Speed Movements in the Context of a Table Tennis Application

Stephan Rusdorf  
Chemnitz University of Technology  
D-09107, Chemnitz, Germany  
+49 371 531 1533  
stephan.rusdorf@informatik.tu-chemnitz.de

Guido Brunnett  
Chemnitz University of Technology  
D-09107, Chemnitz, Germany  
+49 371 531 1533  
brunnett@informatik.tu-chemnitz.de

ABSTRACT
In this paper we summarize the experiences we made with the implementation of a table tennis application. After describing the hardware necessities of our system we give insight into different aspects of the simulation. These include collision detection, physical simulation and some aspects of the design of the virtual opponent.

Since table tennis is one of the fastest sports the synchronization of the player’s movements and the visual output on the projection wall is the most challenging problem to solve. Therefore we analysed the latencies of all subcomponents of our system and designed a prediction method that allows high speed interaction with our application.

Categories and Subject Descriptors
B.8.2 [Performance and Reliability]: Performance Analysis and Design Aids.

General Terms
Algorithms, Measurement, Performance, Design

Keywords
Virtual Reality, Tracking, Latency, Prediction, Collision Detection, Table tennis

1. INTRODUCTION
Most applications for immersive virtual environments are built to allow slow or medium speed user interaction. Examples of this kind of interaction include: changing the position of an object, triggering an action or setting a control parameter. To broaden the range of application for VR systems it seems necessary to integrate technologies that allow faster movements of the user in a virtual environment. This raises the question what response times can be achieved of VR systems built from standard recent hardware components. To answer this question we decided to realize a table tennis simulation because this game involves very fast movements and has moderate space requirements.

For a better understanding of the basic constraints for this application we review a few facts taken from the international rules of table tennis. The ball is specified to have a mass of 2.7 g and a diameter of 4 cm. Its colour may be chose from matt white or orange. The table must be 2.74 m in length, 1.525 m in width and 0.76 m in height. The material of the table is not specified but it has to be guaranteed that the ball bounces back at least 23 cm when it is released at 30 cm height above the table. The bouncing height of the ball is allowed to vary only in a very small range for different positions on the table. While the dimensions of the net are given to be 1.83 m in width and 0.1525 m in height it is interesting to note that there are no such restrictions for the racket. There are no explicit constraints for the size, the shape or weight of the racket but some restrictions for the material that implicitly constrain the mentioned parameters. In a 2003 smashing competition the winner was able to speed up a vertically dropping ball to 31.25 meters per second (69.9 MPH). Of course the maximum speed in a usual table tennis match is much smaller, especially between non-professional players. Our tests showed that even for ambitious non-professional players the ball speed in a match will stay below 20 very often below 10 meters per second.

This paper is organized as follows. After reviewing related work in section 2 we describe two different hardware set-ups that have been used for our application. Here we report on the difficulties we encountered with our initial installation that led us to the development of the current system.

In section 4 we shortly summarize some aspects of our proprietary scene graph library V3D that we use as the basis software for the table tennis application. We also describe the physical laws we implemented to achieve realistic ball trajectories and present important features of our methods for collision detection. Furthermore, we give some aspects of the design of the main game program, especially for the virtual opponent whose behaviour is of crucial importance to achieve an enjoyable game.

In section 5 we deal with the most challenging problem of our application that is to synchronize the player’s movements with the visual output. The development of our solution to this problem proceeds in three steps. First we analyse the latencies of all subcomponents of our system to achieve an estimate for the total delay. Then we suggest a method to predict future movements based on already measured data. The total latency of the system derived in the first step is used as an initial time parameter for the prediction method. In a third step we adjust this parameter in the prediction method to achieve synchronicity of movements and visual output in different test situations.
2. PREVIOUS WORK

In the field of computer support sports there were several aspects. In 1999 there was a research paper [2] which discusses computers supported cooperative play. The application was named PingPongPlus. Even if the work doesn’t really compare to our one, it shows, that there are lot of interesting opportunities when combining sport with computers. For example the game was extended to include other ideas such as try to hit an object with the ball when playing the same. The visual feedback was realized by projecting directly to the table by using a beamer located above the table. An interesting part of the paper was the implementation of the ball tracking system. Therefore 8 microphones were places on the underside of the table. So ball position can be determined by using the audio signal delay. Furthermore in [3] an major point discusses the importance of the input and output devices for computer supported collaborative sports. So not all sports are appropriated for computer support and some of them needs to change the basic idea of the games such as mentioned above.

In [4] it’s clearly concluded that a major aspect of computer sports is the immersion effect of the system. Only when the user “believes” his environment, when he forget that all surrounding object are virtual then the user takes the environment to seriously. This leads to full body activity and with this the user may sweat as he does real sport. To achieve such realistic environments it’s necessary to build a responsive system.

Related work in field of tracking and filtering is given in [5],[6] and [7]. The major points of these papers discusses smoothing and prediction of tracking data. A comparison of prediction filters was presented in [5]. They compare different filtering methods to compensate the system latency especially for orientation prediction of hand movements. They concluded that there is no favorite filter. Results strongly depend on the input data, parameters and requirements. This means for best performance a lot of tests are needed to find the most appropriated parameters. Additionally an adaptive algorithm is suggested, which adapts to the requirements of the application. In paper [6] were shown that, since predicted output signals have more energy, prediction time should kept below 80ms to avoid jittering.

A lot of work has been done in the field of collision detection including papers by Lin and Manocha [8][9][10]. Most recent papers discuss usage of graphics hardware [9]. This approach can only be used for static collision detection. In addition to the geometric data dynamic collision detection system uses information about the motion path of the objects. Usual parameters are velocity and acceleration [11][12].

3. HARDWARE

The first version of our table tennis application was implemented on a Panoram VisCenterII that consists of a curved screen, three analogue Electrohome beamers (model: Marquee8500LC) for front projection and three HP Workstations. The advantage of the system is the DSLS (distributed single logical screen) technique. This feature allows simultaneous usage of 2D and 3D applications distributed over more than one computer displayed on one virtual screen.

For stereoscopic viewing this setup uses active stereo. To allow comfortable high level user interaction we installed an infrared tracking system with three ARTrack1 cameras. The VisCenterII was sufficient for the first tests but involved several problems.

First it should be noted that the HP workstations of this system are outdated. They are no longer supported by HP and only allow the use of OpenGL1.1. Therefore we use an extension of the Chromium library that allows distributed high performance 3D graphics with optional edge blending on Windows and Linux to get similar comfort as with the HP-DSLS.

The curved projection wall is fine for presentations, especially for small groups but has severe drawbacks when interaction is involved. This is because the standard 3D projection methods are only correct for planar projection surfaces. The projection on a curved screen results in a distorted image. Therefore the user can’t see the objects (e.g. the ball) at their real positions which results in collision errors during the simulation.

To remove the distortion we integrated two algorithms into the Chromium package. The first one computes for each polygon the correct vertex positions for the projection on the curved screen. To increase accuracy the sizes of the polygons can be reduced by polygon subdivision. This approach results in a loss of performance for highly complex scenes. The second method uses texture mapping in conjunction with two rendering passes. Here the slowdown is constant and does not depend on the scene complexity but currently is still to high for large texture size. In addition both algorithms currently results in a loss of visual quality.

Further disadvantages of this system include the size of the screen and the fact that front projection is used. Due to the latter a user taller than 1.80m would cast shadows on the screen if he stands closer than about 2 meters to it. Furthermore the screen is only 1.65 in height and has a fixed distance of 40cm from the floor. Since the user mostly looks slightly down the missing image right above the ground leads to a loss of immersion.

To solve these problems our second implementation is based on a rear projection system. For this we use a 4x3m screen that consists of two 2x3m acrylic glass panels with a thickness of 5mm. There are three types of panels available which differ in gain and half angle value. We use the medium panels (gain 2.5, half angle 52 degrees) to achieve the best compromise between visual quality, contrast and viewing angle dependency.

The difficulty with this screen size was the frame construction. We had to build a frame which is able to carry the two panels which are nearly 40kg each. Furthermore the frame should allow a precise adjustment of the two panels to achieve the impression of
a single panel. Here we realized, that a remaining small gap between the panels (approx. 1-2mm) should be darkened by a wire or something similar. The reason for this is that the user sometimes looks through the gap directly to the beamer and the bright light is more bothering than a small dark bar in the middle of the screen.

![Figure 2: rear projection with four tracking cameras](image)

The requirements could be fulfilled by a self made wooden frame. It is dismountable in three hours and portable on a van of 4,20m in length. The reassembling including complete set-up and cabling needs about five hours. An optional construction allows complete darkening of the back side of the screen in order to get best visual quality in rooms with daylight. The two panels are mounted on special brackets to allow exact alignment of the panels.

For the projection we use 2 LCD beamers (model Epson EMP74) with a resolution of 1024x768. This leads to a pixel size of 4x4mm which is not optimal but doesn’t affect our work. The beamers have to be installed 6 meters behind the screen to achieve a screen filling image. To reduce the space requirements we project the image on a mirror that is positioned 4 meters behind the screen and the beamers are located in between the mirror and the screen. In contrast to the VisCenterII this setup uses passive stereo.

There are no high end requirements for the computer. Currently we use a standard pc with 3 GHz PentiumIV processor and a nVidia Quadro or ATI X600 graphics board. This is enough to run the tracking software on the same machine without problems. For real 3D audio output we use a 5.1 sound system.

As before with the curved screen we now use our optical tracking system consisting of three ARTtrack1 cameras in conjunction with the new rear projection. One camera is installed in the center of the top horizontal beam of the frame and the two others sit on the left and right vertical beam of the frame.

The table tennis application requires two targets to be tracked: the polarization glasses and the table tennis racket (fig. 3). When constructing the targets the characteristics of the tracking system have to be considered. The system needs a minimum of 4 markers per target. In order to achieve robust tracking 6 markers are used for the racket. For convenience we let all markers stick out on the same side of the racket. This allows to lay the racket on a table when the game is over. For the glasses we use a standard target construction with five markers.

![Figure 3: the two objects with installed tracking target](image)

In contrast to common VR applications where the user stands relatively still and makes only slow movements with his hands or arms the table tennis sport involves fast, wide range movements. This is an important aspect when it comes to tracking. In such an application the probability of marker occlusion of a tracked object is much higher. If there are not enough cameras installed, this results in a less reliable tracking process. The tracking works reasonably well with three cameras but we achieved considerably more reliable results with four cameras. This situation is shown in figure 2. A fifth camera of course improves the tracking again but the gain in quality that can be achieved with five instead of four cameras is quite small.

4. SOFTWARE

4.1 V3D Basis software

Besides commercial and open source scene graph libraries as WTK or OpenSG we use a proprietary library called V3D [14] for implementing complex 3D applications. V3D consists of a module for the user interface handling and the underlying render engine. For reasons given below the table tennis application is completely based on V3D. Therefore the following paragraph describes some relevant aspects of the system.

![Figure 4: Communication between the different parts of the V3D user interface](image)

V3D is system independent and runs on Windows, Linux or Unix. The most important advantage of our system is that we have full access control of all components. We are able to integrate new algorithms at any level of the library. To achieve high performance it is possible to optimize or exchange single modules for a special purpose.

Another reason, why we prefer to use our interface is the fact that there are already implemented algorithms, which can be used for the table tennis application. For example we mention a novel interaction method that has been presented at VR 2004 [15]. Here the user only needs three thimbles attached to the thumb, the middle and the index finger. The user can now control the
application by static and dynamic gestures without being handicapped by unwieldy input devices.

In addition static and dynamic gestures with a single marker can be used to control the system [16]. Further implemented methods include a basic physics engine, collision detection, tracking algorithms and components for loading and managing complex dynamic scenes with event based animation system.

4.2 Physics

If we consider the mass \( m \) of the ball to be concentrated in its center \( \vec{b}(t) \), the movement of \( \vec{b}(t) \) is governed by Newton’s second law that can be formulated as the following differential equation

\[
\frac{d^2 \vec{b}(t)}{dt^2} = \sum \vec{f}_i = m\ddot{a} = -\frac{\tau}{\tau} m \frac{d}{dt} \vec{b}(t)
\]

Here \( \ddot{a} \) denotes the constant acceleration due to gravity and \( \tau \) is a parameter that describes the influence of friction due to air resistance. This differential equation has the general solution

\[
\vec{b}(t) = \vec{b}_0 + \ddot{a} \tau^2 \frac{t - \tau_0}{\tau} + (\vec{v}_0 - \ddot{a} \tau(1 - e^{-t - \tau_0} / \tau) \tau)
\]

Note, that \( \ddot{a} = (0, -g, 0) \), i.e. the acceleration has zero x and z components.

Since we deal with an extended object we have to include the influence of the spin on the movement of the ball. This influence is called the “Magnus Effect” which causes the so called Magnus force [17]. This force acts perpendicular to the velocity vector and to the spin axis and can be computed as follows.

\[
F_M = C_M \rho D^3 f v
\]

where \( C_M \) is the Magnus coefficient, \( \rho \) is the density of the air (1.26 kgm\(^{-3}\)), \( D \) denotes the diameter of the ball, \( f \) is the spin frequency of the ball and \( v \) is the velocity vector of the ball. Knowing the force on the ball, the acceleration can be calculated from \( F = ma \). Further calculations are necessary to decrease ball spin over time. The decrement depends on the air resistance of the ball, the current velocity and the current rotation speed.

As long as no collision occurs between the ball and some other object in the scene the equations describing the ball movement are evaluated for each frame. If the detection algorithm indicates a potential collision within the time step between the current frame and the next one further evaluations become necessary to compute the point of collision. How this is done will be described in the next section.

To simulate how the ball bounces off after a collision we make use of a heuristic approach described below.

For a collision point \( P \) we denote with \( \vec{V} \) in the incoming velocity of the ball and with \( \vec{N} \) the normal vector of the collision object at \( P \). We compute an initial outgoing velocity vector \( \vec{V}_{out1} \) according to the reflection law.

The magnitude of the velocity \( \vec{V}_{out1} \) and the ball spin after collision also depend on the material properties of the racket and the ball. We implemented a simplified model which uses one parameter to control the length of \( \vec{V}_{out1} \) and another one to set the friction of the material. In a second step \( \vec{V}_{out} \) is modified because
the influence of the ball spin has to be included. For this we compute an axis $A_{\text{effect}}$, which is perpendicular to the normal $N$ and lies in one plane with $N$ and the spin axis. We now determine the rotation speed $v_{\text{effect}}$ with respect to this axis. Then a rotation is applied to the outgoing vector $V_{\text{out}}$, where $A_{\text{effect}}$ is used as the rotation axis and the rotation angle depends on $v_{\text{effect}}$ and the friction of both collision objects. To compute the spin after collision we extract the tangential component from the incoming speed vector $V_{\text{in}}$.

4.3 Collision Detection

To allow interaction a collision detection module has to be integrated into the system. Note, that in the game simulation only six objects are involved: the ball, two rackets, the table, the net and the ground. Furthermore, we can simplify the collision detection by only computing collisions of the ball with the five other objects. All other collisions can be neglected. However, it is necessary to compute all collisions that that occur within the time interval between two frames. For example it is possible that the ball touches both the net and the table within this period. Therefore we need to determine the first intersection that is going to occur in the relevant time interval. Based on this collision the movement of the ball is recomputed and another collision search is started for the remaining time interval.

In the following we describe in more detail the computations performed to determine a collision between the ball and the racket. We associate a bounding box with the racket. Two planes of the box are parallel to the rubber coated surface which is the most important part of the racket. The other planes of the box are chosen in order to minimize the volume of the box. With the center point $O$ of the box we associate a coordinate system $F$ with axes parallel to the edges of the box. Let $b(t)$ denotes the trajectory of the ball’s center. Note, that $b(t)$ can be computed for any time value $t$ based on a prediction method that will be described in section 5.

When the ball comes close to the racket, measured by the distance $|b(t)-O(t)|$, we compute the distances of the ball’s center from the planes bounding the box:

$$d_b(t) = \langle N(t), b(t) \rangle - d(t),$$

where $N(t)$ denotes the normal vector of a face of the box and $d$ it’s orthogonal distance from the origin. If within two frames a sign change for one of these functions occurs we search for an intersection point between the box and the ball. For this we compute the intersection points between the ball and the planes facing the ball and check whether an intersection point lies on the box. Note, that different of such points may exist. In this case we choose the intersection point with the smallest $t$ value. Let $b_o$ denotes the position of the ball’s center and $v_b$ it’s velocity at the time $t_o$ of intersection with the box. We then shoot a ray starting at $b_o$ in the direction of $v_b$ into the box and determine the intersection point with the exact geometry of the racket. Based on this position we approximate the point where ball of radius $r$ touches the racket.

4.4 Significance of the Opponent

The user acceptance of the table tennis application very much depends on the behaviour of the virtual opponent. As in real life an enjoyable game takes only place when both players are nearly on the same level. Therefore the simulation of the opponent is an important aspect of the system.

Since it is not difficult to create an opponent that gets all points one has to think about strategies to reduce the skills of the virtual player. As known from other game simulations (e.g. chess) we realized different skill levels for the opponent that can be chosen by the user. By reducing the level the user takes influence on the following properties of his opponent. First, an upper bound for the maximum speed of his racket is defined. Second, we define a time delay that consists of two components. The first one is a constant value that corresponds to a reaction time. The second component depends on the distance the virtual player has to move from his current position to the new position where he intends to hit the ball. This delay time may cause that the player misses the ball or hits it at a different position as intended. Furthermore, we included a random noise function in the computation of the parameters for an optimal return of the ball. The amplitude of the noise function is decreased for higher game levels but increases over the time the ball is played without interruptions. The last effect intends to model an increasing likeliness of mistakes when the ball is successfully returned for a long time.

4.5 Main Application

The table tennis application has been realized based on our scene graph library V3D. For the rules of the game a special plug-in has been created. To control the set-up of the game we provide intuitive interaction techniques. For example the user can use the second hand to adjust the level of the game with the device mentioned in section 4.1. Reset and Replay can be triggered by certain gestures of the player. Mouse/Keyboard control in still possible.

The virtual environment is completely defined by an external scene that is loaded by the application. This has the advantage that the environment can be easily changed (e.g. replacement of the table) or extended to achieve a nicer visual appearance. A map is
used to assign their game relevant meaning to the important objects of the scene (e.g. object 7 is the ball).

Sound data can be defined in order to give a realistic audio feedback. Currently we use our own implementation of spatial audio calculation which supports Prologic Surround and the Doppler Effect. It is well known that audio feedback is an important aspect to achieve immersion into a virtual environment. For the table tennis we found that the performance of the human player decreases, when audio is disabled.

5. MANAGING HIGH SPEED INTERACTION

As mentioned in the introduction the major difficulty is the synchronization of the players movement with the visual output. We begin by analyzing the latency of our system.

5.1 Latency

We define the total latency of the system as the delay between the movements of the real objects (e.g. the racket) and the corresponding visible feedback at the screen.

To explain why latency is a big problem we note that a racket speed of 15 meters per second (which is very difficult to achieve for a non professional player) and a total latency of 100ms in the system would result in a maximum error of more than one meter in the racket position. This illustrates the need of correction algorithms.

![Figure 8: latency of subcomponents](image)

The total delay is the sum of the latencies of the sub components of the system.

The tracking system is the first element in the chain of components. As we use the ART system we can refer to the measurement protocol [1]. This protocol describes the latency of the ART tracking system depending on different parameters as e.g. the number of cameras. In a situation of four cameras two to four bodies an no single marker tracking the system has a latency of approximately 20ms using a 2.4GHz pc. The variation is about 1ms. This nearly reflects our set-up. Note, that the latency does not depend on the tracking frequency.

The next component is the application itself. This includes the processing of the incoming tracking data and of course the complete visualization with OpenGL. Since the current table tennis scene does not have to much polygons, the delay is very short. This part of the system consumes about 10ms or less of the system’s latency when 100Hz rendering is used. Unfortunately this value is not constant but depends on the rendering frequency and changes from frame to frame. Note that this value is not only determined by the rendering frequency. The reason for this is the OpenGL pipeline. In general to get a high frame rate the use of glFinish() should be avoided. However, if this done, due to the asynchronous behavior of the OpenGL pipeline the duration to the final buffer swap can now vary. In addition to this the vertical synchronization has influence on this timing. To achieve a smoother behavior of the whole system a more constant frame rate is preferred over a higher frame rate that is oscillating. So we determine the actual display frequency and adapt the latency to the corresponding value.

![Figure 9: latency diagram for the tracking system[1]](image)

Additionally we have to consider that there is a delay between the moment the tracking data is received by the application and the beginning of data processing. In our system a separate thread is responsible for providing the tracking data to the application. In this way we can guarantee that each time the application accesses the most recent data without need for a run through a buffer. This delay has to be measured and included in the system’s latency. The value is between zero and the length of a rendering frame and can be exactly measured each time it is needed.

Finally we have to take into account the last device in the visualization pipeline - the beamer. First, we will justify our decision to use LCD beamers. It’s clear that CRT models cannot compare to digital beamers when it comes to brightness, size and price. But why not use a DLP-Model? As already mentioned the system works in passive stereo mode. In the evaluation phase we tested several beamer/filter combinations. We got best visual results with Silverfabric linear High Transmission Polarizers which require LCD beamers. We found that this combination has the best price/performance ratio. A DLP-beamer can be slightly faster but would require the use of a different filter system. As the result we would get lower brightness or we had to invest into a more expensive beamer.

To estimate the beamer latency we connected the beamer and a CRT monitor to the same output of the computer by using a VGA splitter. A test program projects a moving vertical bar onto a linear frame scale. The bar moves one unit per frame. In the evaluation phase we tested several beamer/filter combinations. We got best visual results with Silverfabric linear High Transmission Polarizers which require LCD beamers. We found that this combination has the best price/performance ratio. A DLP-beamer can be slightly faster but would require the use of a different filter system. As the result we would get lower brightness or we had to invest into a more expensive beamer.
camera nearly corresponds to the frame rate. Now we can compare the positions of the bars. Figure 10 shows the comparison between the beamer and the CRT monitor. The tests show that our beamer is most of the time nearly one frame slower than a CRT monitor. Sometimes the delay is even two frames. Taking into account, that we use frame rates from 60 to 85 Hz the delay is about 20ms (15 to 30ms). This value is not constant but oscillates depending on the real frame rate of the beamers LCD and due to the missing synchronization of the signals. At this time we don’t have a more exact measurement method.

Figure 10: two sections of one photo; the beamer (left) is one frame later than the CRT

Counting all terms together results in an approximate total latency of 50ms (20+20+10) plus the time defined by the age of tracking data (which is measured every frame) for the complete system with the current set-up.

5.2 Prediction
In order to cope with the latency the system has to predict the movements of the tracked objects (racket, glasses) over this duration. It is not enough to predict over the 20ms of tracking latency, because the user does not interact with the internal simulation of other objects (e.g. the ball) but with the visible image. So the prediction must take place over the whole pipeline. In [4] several filters for prediction and smoothing of incoming tracking data were discussed. In their conclusion the authors did not prefer any filter and suggested to use application dependent adaptive filtering. Furthermore our tracking system already uses an integrated Kalman Filter for noise reduction of all 6 degrees of freedom. So an additional KF only makes sense, when the integrated KF from the tracking system is disabled. We decided to use the integrated filter for noise reduction of the tracking data and we implemented additional functionality for prediction.

We predict the position for the glasses and the racket based on their velocities and acceleration values in the last three measurements. This seems to be a very suitable approach for our intention because this reflects the real behavior of the racket or the head. In addition to this the algorithm takes into account some simple movement rules for the human body. For instance it is not possible to move the hand out of the range of the arm or to rotate the hand at any angle. For this constraints the head position is included in the calculations. To minimize the prediction error a maximum velocity and acceleration for translation and rotation can be set. Such properties help to remove peaks in the prediction system.

Our approach results in a very responsive prediction which tends to jitter a bit. The major reason for this is not the tracking error, but the variation of the latency from one frame to another. If the prediction time is not precise the prediction can’t be too. Some possible aspects for improvements will be discussed in the last section.

Note that the only important moment we need precise position and orientation data of the racket is the time of collision with the ball. We do not need to have exact prediction for every frame. This is an advantage of the special table tennis application because our tests show that the time of collision is approximately in the middle of a more or less smooth movement of the racket. This supports the assumptions used by the prediction algorithm. However, to predict the very fast movements of professional players will become more difficult. For this purpose it will be necessary to improve the tracking rate and the latency.

In addition to the prediction of the racket’s position and orientation the prediction is used for the head tracking too. This is less difficult because of the smoother movements and lower speed of the head.

5.3 Adjusting Prediction and Latency
The prediction method only approximates the movement of an object and the prediction interval is approximated based on the latency estimation. Therefore it is necessary to adjust these two approximations to achieve reasonable results. This is done based on the following simple procedure. A person equipped with tracked glasses stands in front if the screen and holds up a tracked object (e.g. the racket). When the object is moved it can be visually observed whether the latency preset (that is used for prediction) is too small or too large. If the virtual object on the screen follows behind the real object the latency has to be increased. If the virtual object is ahead of the real one the latency has to be decreased. If both objects stay congruent during the movement a reasonable approximation for the latency has been found.

As the initial value we used our estimation of 50ms for the total latency plus a varying value that corresponds to the age of the tracking data. As explained in section 5.1 this additional value that varies between zero and the duration of one frame (or one
instance of tracking data) can be exactly determined. In figure 12 we show the change of this value over the simulation with a display frequency of 100 Hz and a tracking frequency of 60 Hz. Using a latency preset of 50ms the resulting total latency varies between 50ms and 67ms, i.e. the maximal age of data is 17ms. Based on our experiments we found that the estimated latency of 50ms is close enough to serve as a good initial value for our simple adjustment procedure. A few iterations of this procedure are usually sufficient to fine tune the latency value for optimal results.

Using a latency preset of 50ms the resulting total latency varies between 50ms and 67ms, i.e. the maximal age of data is 17ms. Based on our experiments we found that the estimated latency of 50ms is close enough to serve as a good initial value for our simple adjustment procedure. A few iterations of this procedure are usually sufficient to fine tune the latency value for optimal results.

6. RESULTS AND PERSPECTIVES

With our implementation we have shown, that the high requirements of a table tennis simulation can be met with recent techniques for hardware and software especially for tracking and prediction.

We presented the system at different events as such as IEEE VR2005 in Bonn, Germany. In most cases we got very positive feedback.

Users with no experience with VR environments can use our system without any introduction. The reason for this is the closeness of our simulation to a real table tennis environment. This is achieved by the realistic video and audio output and the familiar input device. The complete environment is immediately accepted by the users. Users who are familiar with VR environments are mostly impressed by the fast response of the system. The third category include the table tennis players. They confirmed the nearly realistic behavior of the system with respect to the ball motion including the influence of spin. However, since they perform much faster movements than other users the limits of the simulation with respect to correct prediction become obvious.

We realized that the sensation of immersion created by our system is very high. Very often we noticed that users tried to put the racket on the table after finishing the game. Sometimes very enthusiastic players hit the screen with the racket. These incidents indicate that the player seems to forget what the environment really looks like.

We further conclude that the frequency of the tracking system is not the major bottleneck of the system. Higher tracking rates certainly improve the result but a more important point is latency. Incrementing the tracking rate to 100Hz will not improve the system behavior as much as decreasing the latency to 20ms. Nevertheless a higher tracking rate would be preferred especially for the high speed requirements of a trained table tennis player. As described in the introduction the ball speed and the racket speed too can reach more than 110 kilometers per hour (30m/s). Using 60Hz tracking rate this results in 50cm measurement point distance per frame. There are tracking system available supporting 200Hz tracking rate and 10 milliseconds latency (described in the datasheet). Unfortunately we couldn’t test such a system so far but this would of course improve responsiveness and accuracy of the system.

Further developments will concern the distribution of our application to allow two remote human users to play against each other. For this it will be necessary to cope with the network latency. Furthermore, we plan to realize a table tennis simulation with multi-user support on a single screen similar to [13]. This will allow us to realize table tennis doubles. We also intend to integrate a haptic feedback for the ball collision with the racket in our simulation.

7. ACKNOWLEDGMENTS

We want to thank Advanced Realtime Tracking for the good collaboration. Special thanks to Steffen Seeger for fruitful discussions on the physical simulation.
8. REFERENCES


